**CSE 3461**

**Lab2: RDT 3.0 over UDP Server and Client**

**Name: Tianyuan Li**

**ID: 200450430**

**Compiling code:**

1. Open a terminal and change to the directory where server.c and client.c files are.

2. Type “make” in the command line.

3. Open another terminal and change to the same directory.

3. Type “./server 3000 <loss probability> <rdt>” in one terminal. (loss probability and rdt are optional depending on which task you want to test.)

4. Type “./client 127.0.0.1 3000 <filename> <loss probability> <rdt>” in the other terminal. (loss probability and rdt are optional depending on which task you want to test.))

5. The procedure of the transportation will be showing in the terminal on both sizes.

6. When it is done, there will be a new file called “copy<filename>”. Now, you can test if the two files are identical or not by entering “diff <filename> copy<filename>” in the terminal.

**Description:**

The lab is to have reliable data transfer (RDT 3.0) built on UDP. For task 1, the code could parse the file data into packets (1K each) and transport packets without loss. For task 2, when the loss probability is added, a random float number is generated and compared to the loss probability. If the random number is larger, then the client side will accept the packet. Otherwise, it will drop the packet. At the end, it will show how many packets out of total number of packets are accepted. For task 3, we follow the algorism of RDT 3.0 to set up a timer when waiting for an ack at the server side. In addition, there is a probability of unsuccessfully sending ack to server on the client side. The probability is set the same as packet loss probability. When testing the code, the server side will show expected ack, timeout, the packet retransmitted and total retransmission times. After 10 times of timeout, the server will assume that the client break down the connection and shut down the process also.

**Result:**

The code worked perfectly and successfully finished all 3 tasks.

**Difficulties:**

Debugging the code was a nightmare again. For task 1, there are no difficulties setting up UDP. However, when I added header information to the packet data, I should increase a little bit of the buffer size. For task 2, I had a difficult time generate a random float number between 0 and 1. When I enter 0.4 as the loss probability in the terminal, whatever the float number generated randomly was always larger. Finally, I found out that when I assigned the fifth argument in the terminal to “loss”, I used atoi rather than atof, so when I entered every number less than 1, it would be 0. For task 3, I spent about a week on this. Adding a timer and ack on the server side wasted a lot of time repeat testing. After finishing the first 2 task, task 3 was not as difficult as I thought. It was just more things to pay attention. It would be better to find out where the code went wrong by printing out as many messages as I could evert step.